HEURYSTYKA Z REGUŁAMI PRIORYTETOWYMI DLA PROBLEMU HARMONOGRAMOWANIA PROJEKTU Z OGRANICZONYMI ZASOBAMI

Marcin KLIMEK, Piotr ŁEBKOWSKI

Streszczenie: W artykule analizowane jest zagadnienie harmonogramowania projektu z ograniczonymi zasobami RCPSP (ang. *Resource-Constrained Project Scheduling Problem*). Opracowano heurystyki: jednoprzebiegową i wieloprzebiegową, które tworzą rozwiązania na podstawie reguł priorytetowania zadań. Porównano skuteczność kilkunastu reguł priorytetowych, przy zastosowaniu równoległego i szeregowego schematu generowania harmonogramu wstecz i w przód. Testy przeprowadzono przy użyciu instancji testowych z biblioteki PSPLIB (*Project Scheduling Problem LIBrary*).

Słowa kluczowe: harmonogramowanie projektu z ograniczonymi zasobami, reguły priorytetowe, harmonogramowanie w przód i wstecz, równoległy i szeregowy schemat generowania harmonogramu

1. Wprowadzenie

Problem harmonogramowania projektu z ograniczoną dostępnością zasobów RCPSP jest jednym z ważnych i często analizowanych zagadnień optymalizacyjnych. Jako funkcje celu dla RCPSP najczęściej rozważane są kryteria czasowe tj. minimalizacja czasu trwania projektu lub kryteria ekonomiczne tj. maksymalizacja sumy zdyskontowanych przepływów pieniężnych [2]. W tym artykule rozpatrywane jest zagadnienie minimalizacji czasu trwania projektu (*makespan*).

Problem harmonogramowania projektu z ograniczonymi zasobami jest zagadnieniem silnie NP-trudnym [1]. Ze względu na złożoność problemu dla RCPSP stosowane są:

- algorytmy dokładne – za ich pomocą znajdowane są harmonogramy optymalne, ale ze względu na wykładniczą złożoność obliczeniową mogą być wykorzystywane jedynie dla projektów o mniejszej liczbie zadań (dla projektów wielozadaniowych czas obliczeń często jest zbyt długi),
- algorytmy heurystyczne, przybliżone – dla projektów wielozadaniowych.

W praktyce często liczba zadań jest duża i wskazane jest stosowanie procedur heurystycznych, które nie dają gwarancji znalezienia harmonogramów optymalnych. Wśród heurystyk wyróżnia się algorytmy konstrukcyjne oraz procedury lokalnych poszukiwań LS (ang. *Local Search*).

Algorytmy konstrukcyjne generują najczęściej jedno rozwiązanie (są jednoprzebiegowe ang. *single-pass*) na podstawie prostych reguł priorytetowania (algorytmy priorytetowe) lub wstawiania czynności (algorytmy wstawień). Ich główną zaletą jest szybki czas działania. Harmonogramy wygenerowane za pomocą procedur konstrukcyjnych są na ogół gorszej jakości do harmonogramów tworzonych za pomocą algorytmów lokalnych poszukiwań. Algorytmy konstrukcyjne są jednak przydatne np. do tworzenia rozwiązań inauguracyjnych dla bardziej skutecznych wieloprzebiegowych (ang. *multi-pass*) procedur LS tj. algorytmy

genetyczne, symulowanego wyżarzania, przeszukiwanie z zakazami, algorytmy genetyczne itd. Analiza efektywności różnych procedur dla problemu RCPSP przedstawiona jest w pracach przeglądowych [3,8-9].

W tej pracy analizowany jest algorytm priorytetowy dla problemu RCPSP z minimalizacją czasu trwania projektu. Poza standardową procedurą generującą jedno rozwiązanie na podstawie danej reguły priorytetowej opracowany jest algorytm wieloprzebiegowy. W celu ilustracji analizowanego zagadnienia i proponowanych heurystyk wykorzystany jest przykładowy projekt złożony z 8 zadań. Analiza eksperymentalna opracowanych procedur i reguł priorytetowych przeprowadzona jest przy wykorzystaniu instancji testowych z biblioteki PSPLIB [10].

2. Opis problemu

Projekt (przedsięwzięcie) to unikalny zbiór współzależnych czynności (zadań) wykonywany przy użyciu dostępnych zasobów (pracowników, maszyn). Zadanie stanowi wyodrębnioną część projektu, dla którego do realizacji wymagana jest określona ilość czasu i zasobów. W tej pracy rozważany jest problem harmonogramowania projektu, w którym szukany jest wektor czasów rozpoczęcia czynności niepodzielnych (ang. *nonpreemptive*), z jednym sposobem ich realizacji (*single-mode RCPSP*), wykonywanych przy użyciu ograniczonych, odnawialnych zasobów (pracowników, maszyn). Jako funkcja celu stosowana jest minimalizacja czasu trwania projektu (ang. *makespan minimisation*) (patrz: wzór 1) przy uwzględnieniu ograniczeń zasobowych (patrz: wzór 2) i kolejnościowych (patrz: wzór 3) typu koniec-początek bez zwłoki (ang. *finish-start, zero-lag precedence*), w których następnik może rozpocząć się bez zwłoki po zakończeniu poprzednika.

Minimalizacja F:

$$F = ST_{\mathrm{N+1}} \tag{1}$$

Przy następujących ograniczeniach:

$$\sum_{i \in J(t)} R_{ik} \le A_k, \qquad \forall t = 1,\ldots,ST_{N+1}, \forall\, k = 1,\ldots,K \tag{2}$$

$$ST_i + D_i \le ST_j \qquad \forall (i,j) \in E \tag{3}$$

gdzie:

N – liczba zadań projektowych,
i – indeks (numer) czynności, i = 1, …, N,
ST_i – czas rozpoczęcia zadania i,
$J(t)$– zbiór zadań wykonywanych w okresie [t-1, t] ,
K – liczba typów zasobów,
k – indeks (numer) typu zasobu, k = 1, …, K,
R_{ik} – zapotrzebowanie zadania i na zasób typu k,
A_k – dostępność zasobów typu k,
D_i – czas trwania czynności i,
E – zbiór łuków opisujących relacje kolejnościowe między zadaniami w grafie projektowym $G(V, E)$ – projekty przedstawiane są w reprezentacji sieci czynności *AON* (ang. – *Activity-On-Node*) jako graf skierowany połączonych węzłów (zbiór V) reprezentujących zadania.

Przy harmonogramowaniu zadań z wykorzystaniem reguł priorytetowych wykorzystywane są dane z analizy czasowej problemu tj.:

- ES_i (ang. *Earliest Start*) – najwcześniejszy możliwy czas rozpoczęcia zadania *i*,
- EF_i (ang. *Earliest Finish*) – najwcześniejszy możliwy czas zakończenia zadania *i*,
- LS_i (ang. *Latest Start*) – najpóźniejszy możliwy czas rozpoczęcia zadania *i*,
- LF_i (ang. *Latest Finish*) – najpóźniejszy możliwy czas zakończenia zadania *i*.

Najwcześniejsze możliwe czasy rozpoczęcia ES_i i zakończenia EF_i zadań wyznaczane są rekurencyjnie kolejno dla czynności od 0 do N+1 przy uwzględnieniu zależności kolejnościowych między zadaniami:

$$ES_0 = 0 \tag{4}$$

$$ES_i = \max_{j \in ZP_i}(ES_j + D_j), \qquad \forall i \in V \tag{5}$$

$$EF_i = ES_i + D_i, \qquad \forall i \in V \tag{6}$$

gdzie:

ZP_i - zbiór bezpośrednich poprzedników zadania *i*.

Najpóźniejsze możliwe czasy rozpoczęcia LS_i i zakończenia LF_i, powinny uwzględniać nieprzekraczalny czas zakończenia projektu *DD* (ang. *due date*) oraz minimalny czas realizacji wszystkich następników (nie tylko bezpośrednich) danej czynności. Analiza czasowa prowadzona jest rekurencyjnie w kolejności od zadania N+1 do zadania 0:

$$LS_{N+1} = \max(ES_{N+1}, DD) \tag{7}$$

$$LF_i = \min(\min_{j \in ZN_i}(LF_j - D_j), EF_i + D_i), \qquad \forall i \in V \tag{8}$$

$$LS_i = LF_i - D_i, \qquad \forall i \in V \tag{9}$$

gdzie:

ZN_i - zbiór bezpośrednich następników zadania *i*.

Rozwiązaniem problemu RCPSP z minimalizacją czasu trwania projektu w reprezentacji bezpośredniej jest wektor czasów rozpoczęcia zadań. Reprezentacja bezpośrednia nie jest często używana w heurystykach m.in. ze względu na trudności w przeszukiwaniu przestrzeni potencjalnych rozwiązań. W procedurach LS stosowane jest kodowanie harmonogramów przy wykorzystaniu reprezentacji [9]:

- listy czynności (ang. *activity list*) – rozwiązanie to permutacja numerów kolejnych zadań przy uwzględnieniu relacji kolejnościowych (najbardziej efektywne i najczęściej stosowane kodowanie dla RCPSP [9]),
- reguł priorytetu – rozwiązanie to lista priorytetowa, w której każdemu zadaniu przypisana jest reguła priorytetu,
- wektora opóźnień – rozwiązanie to wektor opóźnień poszczególnych czynności od najwcześniejszego możliwego ich czasu rozpoczęcia.

Rozwiązania w reprezentacji listy czynności lub reguł priorytetu dekodowane są przy użyciu schematów generowania harmonogramu SGS (ang. *Serial Schedule Generation Scheme*) w realizowalne uszeregowania, uwzględniające ograniczenia czasowe i zasobowe, w reprezentacji bezpośredniej. Dla deterministycznego RCPSP wykorzystywane są [6]:

- szeregowy SGS (ang. *serial SGS*) – procedura dekodująca, w której w kolejnych chwilach t wyznaczany jest czas rozpoczęcia dla pierwszego nieuszeregowanego zadania z listy czynności lub priorytetowej, w najwcześniejszym możliwym czasie przy uwzględnieniu ograniczeń kolejnościowych i zasobowych,
- równoległy SGS (ang. *parallel SGS*) – procedura dekodująca, w której w kolejnych chwilach t rozpoczynane są wszystkie nieuszeregowane zadania (analizowane w kolejności wynikającej z listy czynności lub priorytetowej), które mogą być rozpoczęte w chwili t przy uwzględnieniu ograniczeń kolejnościowych i zasobowych.

Przy ustalaniu czasów rozpoczęcia zadań stosowane są różne strategie:

- harmonogramowanie w przód (ang. *forward scheduling*) – planowanie kolejnych zadań „od początku" listy czynności lub priorytetowej w najwcześniejszych możliwych terminach przy uwzględnieniu ograniczeń kolejnościowych i zasobowych,
- harmonogramowanie wstecz (ang. *backward scheduling*) – planowanie kolejnych zadań „od końca" listy czynności lub priorytetowej, przy uwzględnieniu ograniczeń kolejnościowych, zasobowych, rozpoczynając planowanie wstecz od terminu przyjętego zakończenia projektu (*due date*).

3. Heurystyka z regułami priorytetowania zadań

Algorytm priorytetowy tworzy harmonogram na podstawie priorytetów poszczególnych zadań wyznaczanych dla przyjętych reguł priorytetowych. Generowana jest lista czynności uwzględniająca te priorytety, którą procedura dekodująca SGS zamienia w harmonogram w reprezentacji bezpośredniej (ustalane są czasy rozpoczęcia zadań).

Heurystyki działające w oparciu o listę priorytetową czynności tworzą harmonogramy, które mogą znacznie odbiegać od optymalnych, ale są przydatne jako rozwiązania inauguracyjne w procedurach poprawy tj. metaheurystyki symulowanego wyżarzania.

Klasyczny algorytm priorytetowy jest jednoprzebiegowy – tworzy jedno rozwiązanie. Autorzy proponują heurystykę wieloprzebiegową, która generuje większą liczbę potencjalnych uszeregowań przy uwzględnieniu danej reguły priorytetowej [4-5]. Jest to zrandomizowana wersja klasycznego algorytmu priorytetowego, w której wprowadzona jest losowość w budowaniu list czynności na podstawie priorytetów zadań (koncepcja zbliżona do selekcji osobników metodą rankingowej w algorytmach genetycznych [11]).

W pojedynczym przebiegu proponowanej heurystyki wieloprzebiegowej tworzona jest lista czynności L, która jest następnie dekodowana przy użyciu procedury SGS. Przy ustalaniu listy L wykorzystywane są informacje:

- z listy LP – statycznej listy czynności wyznaczonej na podstawie priorytetów zadań wyznaczonych dla stosowanej reguły priorytetowej (kolejność na liście LP taka jak ustalona dla danej reguły w klasycznym algorytmie priorytetowym jednoprzebiegowym),
- z listy LA – dynamicznej listy dostępnych zadań, nie dodanych jeszcze do listy L, tych które mogą być już rozpoczęte uwzględniając zależności kolejnościowe (poprzednikami ich są zadania znajdujące się na liście L).

Kolejne kroki algorytmu wieloprzebiegowego wyglądają następująco [4-5]:

Krok 1:

Ustawienie pustej listy L. Na liście LA wpisanie wszystkich następników zadania o numerze 0 posortowanych w porządku takim jak na liście LP.

Krok 2:

Wylosowanie z listy *LA* jednego zadania *i* przy uwzględnieniu „szans" poszczególnych zadań wyznaczonych na podstawie ich pozycji na liście *LA*: zadanie pierwsze ma największą liczbę szans na selekcję równą liczbie zadań na liście *LA*, …, zadanie ostatnie ma przyznaną jedną szansę selekcji (podobnie jak w selekcji rankingowej w algorytmach genetycznych [11]).

Krok 3:

Usunięcie wylosowanego zadania *i* z listy *LA* i dodanie go do listy *L*. Umieszczenie na liście *LA* wszystkich następników zadania *i*, których wszystkie poprzedniki umieszczone są na liście L, z zachowaniem porządku takiego jak na liście *LP*.

Krok 4:

Powtórzenie kroków 2-3 aż do wypełnienia listy *L* wszystkimi zadaniami projektowymi.

Wyznaczone listy czynności *L* w poszczególnych przebiegach różnią się od siebie, ale uwzględniają „hierarchię" zadań ustaloną na podstawie danej reguły priorytetowej. Wynikiem działania heurystyki wieloprzebiegowej jest rozwiązanie o najmniejszym czasie trwania projektu, wyznaczone przy wykorzystaniu procedur SGS dla list *L* z kolejnych przebiegów algorytmu.

Znalezienie odpowiednich reguł priorytetowych, może mieć wpływ na jakość uzyskiwanych harmonogramów, zarówno heurystyki jedno- jak i wieloprzebiegowej. W tabeli 1 przedstawione są reguły priorytetowania (reguły P_1-P_{11}), wykorzystywane w badaniach dotyczących problemu *RCPSP* [7], które są analizowane w tej pracy. Dla celów porównawczych dodano regułę P_0, w której poszczególnym czynnościom przypisane są losowe priorytety.

Tab. 1. Reguły priorytetowania zadań [4-5]

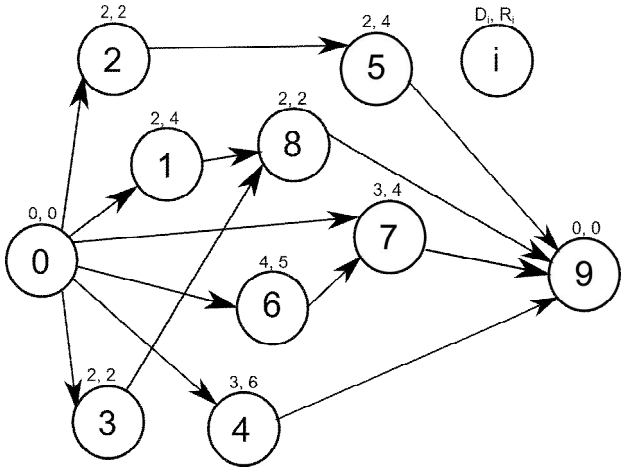
Reguła	Wzór na priorytet zadania *i*	Opis reguły
P_0	$P_0(i) = random()$	losowe priorytety czynności
P_1	$P_1(i) = -ES_i$	minimalny najwcześniejszy czas rozpoczęcia czynności
P_2	$P_2(i) = -LS_i$	minimalny najpóźniejszy czas rozpoczęcia czynności
P_3	$P_3(i) = -LF_i$	minimalny najpóźniejszy czas zakończenia czynności
P_4	$P_4(i) = -EF_i$	minimalny najwcześniejszy czas zakończenia czynności
P_5	$P_5(i) = -(LS_i - ES_i)$	minimalna różnica między najpóźniejszym a najwcześniejszym możliwym czasem rozpoczęcia czynności
P_6	$P_6(i) = -(LF_i - EF_i)$	minimalna różnica między najpóźniejszym a najwcześniejszym możliwym czasem zakończenia czynności
P_7	$P_7(i) = \#N_i$	maksymalna liczba wszystkich następników czynności

P_8	$P_8(i) = \#ZN_i$	maksymalna liczba wszystkich bezpośrednich następników czynności
P_9	$P_9(i) = -D_i$	najkrótszy czas trwania czynności
P_{10}	$P_{10}(i) = D_i + \sum_{j \in N_i} D_j$	maksymalna suma czasów trwania danej czynności i jej wszystkich następników
P_{11}	$P_{11}(i) = D_i \cdot \sum_{k=1}^{K} R_{ik} + \sum_{j \in N_i} D_j \cdot \sum_{k=1}^{K} R_{jk}$	maksymalna suma iloczynu czasu trwania i zasobochłonności danej czynności i jej wszystkich następników

Jeśli stosując daną regułę priorytetową czynności mają takie same priorytety, to na wcześniejszych pozycjach listy czynności umieszczane są czynności oznaczone niższym numerem.

4. Przykład ilustracyjny

Przykładowy projekt przedstawiony jest jako sieć czynności AON na rysunku 1. Składa się z 8 zadań (węzły 0 i 9 to zadania pozorne o zerowym czasie trwania i zapotrzebowaniu na zasoby, reprezentujące początek i koniec grafu). Jest realizowany za pomocą jednego typu zasobu o dostępności równej 10 (A = 10). Na węzłach zdefiniowane są zadania i ich czasy trwania i zapotrzebowanie na zasoby. Krawędzie reprezentują zależności kolejnościowe między zadaniami. Nieprzekraczalny termin realizacji projektu wynosi 14 (DD = 14).

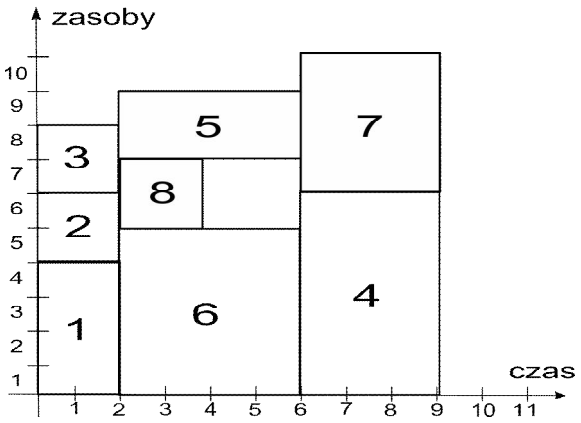


Rys. 1. Sieć czynności *AON* przykładowego projektu

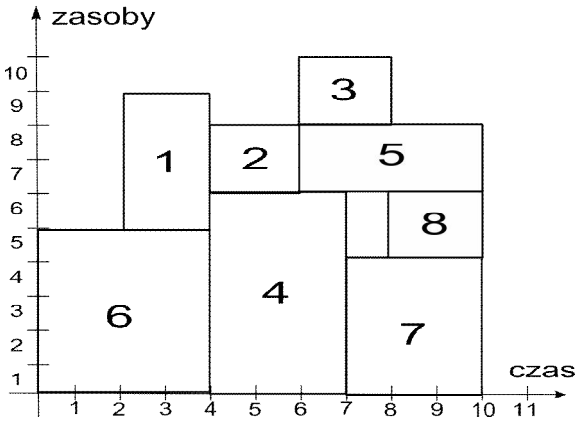
W analizowanych w pracy heurystykach priorytetowych generowana jest lista czynności, ustawionych w kolejności wynikającej z przyjętej reguły priorytetowej, która następnie jest dekodowana za pomocą SGS. W poniższej analizie stosowana jest reguła priorytetowa P_7, w której zadania są w porządku malejącym na podstawie liczby wszystkich następników czynności (poza pozornymi). Priorytety poszczególnych zadań wynoszą: $P_7(1) = 1$, $P_7(2) = 1$, $P_7(3) = 1$, $P_7(4) = 0$, $P_7(5) = 0$, $P_7(6) = 1$, $P_7(7) = 0$, $P_7(8) = 0$.

Na podstawie priorytetów lista czynności jest następująca: {1, 2, 3, 6, 4, 5, 7, 8}. Ta lista jest dekodowana w realizowalny harmonogram (wektor czasów rozpoczęcia zadań), przy uwzględnieniu ograniczeń kolejnościowych i zasobowych, za pomocą procedur SGS. Zadania mogą być rozpatrywane od początku listy czynności {1, 2, 3, 6, 4, 5, 7, 8} (harmonogramowanie w przód) lub od jej końca {8, 7, 5, 4, 6, 3, 2, 1}. (harmonogramowanie wstecz). W przypadku harmonogramowania wstecz generowanie harmonogramu zaczyna się „od końca": od terminu przyjętego zakończenia projektu (w analizowanym przykładzie od *DD* = 14). Po utworzeniu całego uszeregowania, czasy rozpoczęcia zadań są korygowane w ten sposób aby czas rozpoczęcia projektu był równy 0.

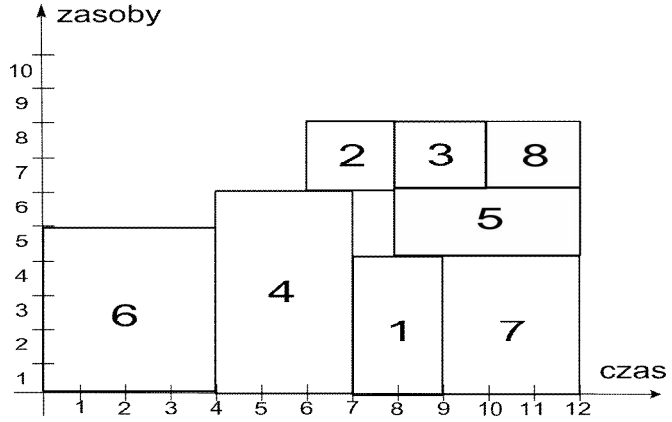
Harmonogramy znalezione dla listy czynności{1, 2, 3, 6, 4, 5, 7, 8} znajdują się na rysunku 2 (harmonogramowanie w przód, szeregowa i równoległa procedura SGS), rysunku 3 (harmonogramowanie wstecz, szeregowa procedura SGS), oraz rysunku 4 (harmonogramowanie wstecz, równoległa procedura SGS).



Rys. 2. Harmonogram znaleziony dla listy czynności {1, 2, 3, 6, 4, 5, 7, 8}, przy harmonogramowaniu w przód dla szeregowej i równoległej procedury SGS



Rys. 3. Harmonogram znaleziony dla listy czynności {1, 2, 3, 6, 4, 5, 7, 8}, przy harmonogramowaniu wstecz dla szeregowej procedury SGS



Rys. 4. Harmonogram znaleziony dla listy czynności {1, 2, 3, 6, 4, 5, 7, 8}, przy harmonogramowaniu wstecz dla równoległej procedury SGS

Wybór procedury dekodowania SGS ma wpływ na jakość uzyskiwanych rozwiązań. Zasadne jest sprawdzenie skuteczności różnych SGS dla analizowanego problemu. W rozpatrywanym przykładzie najlepszy harmonogram o czasie realizacji równym 9 znaleziony jest przy użyciu harmonogramowania w przód (szeregowej i równoległej procedury SGS).

Dla heurystyki wieloprzebiegowej wyjaśnienia wymaga przede wszystkim krok 2. W kroku tym losowane jest jedno zadanie z listy *LA* zadań dostępnych do harmonogramowania zadań. Dla analizowanego przykładu z przyjętą reguła priorytetową i strategią harmonogramowania w przód w kroku 1 do listy *LA* dodawane są następniki zadania 0 w kolejności takiej jak na liście *LP* = {1, 2, 3, 6, 4, 5, 7, 8}. Przy takim założeniu *LA* = {1, 2, 3, 4}. W kroku 2 zadanie jest losowane z listy {1, 1, 1, 1, 2, 2, 2, 3, 3, 4}, na której każde z zadań ma szansę wyboru wynikającą z pozycji na aktualnej liście *LA*: zadanie pierwsze (w tym przypadku zadanie 1) na liście *LA* ma 4 szanse, …, zadanie ostatnie (w tym przypadku zadanie 4) ma jedną szansę wyboru. W kroku 3 zadanie wylosowane w kroku 2 jest wstawiane do listy czynności *L* a jego następniki są umieszczane na liście *LA* w porządku z listy *LP*. Kroki 2-3 są powtarzane aż do umieszczenia wszystkich zadań projektowych na liście *L*. Przy takiej strategii tworzenia listy czynności *L* uwzględnione są priorytety poszczególnych zadań.

5. Wyniki eksperymentów

Eksperymenty przeprowadzono przy wykorzystaniu aplikacji zaimplementowanej w języku C# w środowisku Visual Studio.NET dla projektów testowych z biblioteki PSPLIB [10] ze zbioru J30 (480 instancji 30-zadaniowych) oraz J90 (480 instancji 90-zadaniowych).

Celem eksperymentów jest sprawdzenie efektywności proponowanych heurystyk priorytetowych (jednoprzebiegowej i wieloprzebiegowej), znalezienie najlepszych reguł priorytetowych (spośród reguł P_0-P_{11}) i procedur dekodujących SGS (równoległej lub szeregowej) harmonogramowania w przód lub wstecz. W zrandomizowanej heurystyce wieloprzebiegowej dla każdej instancji problemu liczba przeprowadzonych przebiegów wynosi 100 (liczba sprawdzanych rozwiązań jest równa 100). Wyniki obliczeń

zaprezentowane są w tabelach 2 i 3. Dane dotyczące heurystyki jednoprzebiegowej znajdują się w wierszach oznaczonych P_0-P_{11} a wersji zrandomizowanej, wieloprzebiegowej w wierszach oznaczonych rP_0-rP_{11}.

Tab. 2.Wyniki eksperymentów obliczeniowych dla zestawu J30

	Schemat generowania harmonogramu											
	Szeregowy w przód			Równoległy w przód			Szeregowy wstecz			Równoległy wstecz		
	a	*b*	*c*	*a*	*b*	*c*	*a*	*b*	*c*	*a*	*b*	*c*
P_0	68,1	15,5%	155	65,1	10,3%	153	67,0	13,5%	123	68,0	15,2%	74
rP_0	60,8	3,0%	278	60,5	2,6%	246	60,2	2,1%	322	66,3	12,5%	88
P_1	64,7	9,7%	149	64,0	8,4%	155	62,6	6,1%	238	67,3	14,0%	79
rP_1	60,8	3,1%	268	60,6	2,7%	244	60,1	1,8%	326	66,4	12,6%	88
P_2	62,3	5,6%	246	62,0	5,1%	213	67,3	14,1%	67	68,0	15,2%	74
rP_2	**60,2**	**2,0%**	**321**	**60,2**	**2,0%**	**264**	60,4	2,3%	302	66,3	12,5%	87
P_3	62,7	6,2%	240	61,9	4,8%	206	65,8	11,6%	102	67,8	15,0%	75
rP_3	60,2	2,1%	317	60,2	2,1%	257	60,2	2,0%	310	66,4	12,5%	88
P_4	66,2	12,2%	137	64,3	9,1%	151	62,5	5,9%	241	67,3	14,1%	80
rP_4	61,0	3,3%	269	60,6	2,8%	237	**60,0**	**1,8%**	**336**	66,4	12,6%	87
P_5	65,1	10,4%	198	63,1	7,0%	196	71,9	21,9%	50	68,6	16,2%	66
rP_5	60,4	2,4%	309	60,3	2,2%	259	60,6	2,8%	285	**66,4**	**12,5%**	**88**
P_6	65,7	11,4%	198	63,2	7,2%	195	71,8	21,8%	49	68,6	16,3%	66
rP_6	60,4	2,5%	311	60,4	2,3%	255	60,6	2,7%	288	66,4	12,6%	87
P_7	63,2	7,2%	213	62,2	5,5%	190	65,8	11,5%	114	67,8	14,9%	75
rP_7	60,3	2,2%	308	60,3	2,2%	261	60,2	2,1%	314	**66,4**	**12,5%**	**88**
P_8	65,1	10,4%	172	63,4	7,5%	173	65,7	11,4%	140	67,8	14,9%	76
rP_8	60,8	3,0%	282	60,5	2,6%	245	60,3	2,1%	309	66,4	12,6%	88
P_9	69,5	17,9%	137	64,9	9,9%	151	65,6	11,3%	167	67,6	14,7%	78
rP_9	61,1	3,6%	263	60,6	2,8%	243	60,1	1,9%	326	**66,4**	**12,5%**	**88**
P_{10}	62,4	5,9%	234	62,0	5,1%	201	66,7	13,0%	89	67,9	15,1%	74
rP_{10}	60,2	2,1%	314	60,2	2,0%	259	60,3	2,1%	306	66,4	12,5%	87
P_{11}	62,9	6,6%	224	62,2	5,5%	195	67,6	14,6%	82	68,2	15,5%	74
rP_{11}	60,2	2,1%	309	60,2	2,0%	267	60,3	2,1%	305	**66,4**	**12,5%**	**88**

a – średnia wartość funkcji celu *F,*

b – odchylenie średniej wartości funkcji celu *F* od rozwiązania optymalnego,

c – liczba rozwiązań optymalnych (spośród 480 eksperymentów).

Tab. 2.Wyniki eksperymentów obliczeniowych dla zestawu J90

	Schemat generowania harmonogramu											
	Szeregowy w przód			Równoległy w przód			Szeregowy wstecz			Równoległy wstecz		
	a	*b*	*c*	*a*	*b*	*c*	*a*	*b*	*c*	*a*	*b*	*c*
P_0	113,4	19,5%	137	106,5	12,2%	136	107,6	13,4%	188	111,7	17,7%	80
rP_0	102,5	7,9%	261	100,2	5,6%	221	100,1	5,4%	316	107,2	12,9%	90
P_1	106,9	12,6%	132	105,3	10,9%	134	101,1	6,4%	300	110,7	16,6%	82
rP_1	102,4	7,9%	235	100,6	6,0%	212	99,3	4,6%	323	107,8	13,5%	85
P_2	100,6	6,0%	**312**	100,4	5,7%	199	109,9	15,8%	63	112,1	18,0%	68
rP_2	**100,0**	**5,4%**	302	**98,9**	**4,2%**	**251**	100,1	5,4%	298	107,1	12,8%	89
P_3	101,1	6,5%	295	100,5	5,9%	194	107,7	13,5%	89	111,9	17,9%	69
rP_3	100,1	5,5%	302	**98,9**	**4,2%**	**251**	99,9	5,2%	309	107,2	12,9%	89
P_4	109,0	14,8%	126	105,7	11,3%	134	100,5	5,9%	316	110,7	16,6%	81
rP_4	102,9	8,3%	222	100,7	6,1%	206	**99,1**	**4,4%**	**330**	107,9	13,7%	85
P_5	105,7	11,3%	244	102,6	8,0%	185	116,0	22,2%	70	112,7	18,8%	65
rP_5	100,8	6,2%	292	99,2	4,5%	247	100,6	6,0%	286	**107,0**	**12,7%**	**90**
P_6	106,1	11,7%	245	102,6	8,1%	184	115,7	21,9%	71	112,8	18,8%	66
rP_6	100,9	6,2%	290	99,3	4,6%	248	100,6	5,9%	284	107,1	12,8%	90
P_7	101,9	7,3%	259	101,2	6,6%	177	107,6	13,4%	131	112,0	18,0%	70
rP_7	100,5	5,8%	293	99,2	4,5%	243	99,9	5,2%	312	107,2	12,9%	90
P_8	106,3	12,0%	175	103,7	9,2%	155	105,9	11,6%	183	111,6	17,5%	72
rP_8	102,0	7,4%	260	100,1	5,4%	229	99,9	5,2%	312	107,2	12,9%	88
P_9	114,5	20,6%	133	106,4	12,1%	135	105,2	10,8%	218	111,4	17,4%	75
rP_9	103,1	8,6%	242	100,3	5,6%	217	99,7	5,0%	319	107,2	12,9%	89
P_{10}	101,1	6,5%	286	100,9	6,2%	187	108,8	14,6%	98	112,1	18,1%	68
rP_{10}	100,3	5,6%	298	99,1	4,4%	242	99,9	5,3%	310	107,1	12,8%	89
P_{11}	101,6	7,1%	277	101,0	6,3%	183	109,8	15,7%	105	112,3	18,3%	66
rP_{11}	100,4	5,7%	291	99,2	4,4%	245	100,1	5,4%	303	107,1	12,8%	89

a – średnia wartość funkcji celu *F,*
b – odchylenie średniej wartości funkcji celu *F* od najlepszego znanego rozwiązania,
c – liczba rozwiązań identycznych z najlepszym znanym rozwiązaniem (spośród 480 eksperymentów).

Zrandomizowana wersja heurystyki priorytetowej wieloprzebiegowej jest skuteczniejsza od jednoprzebiegowej dla każdej z reguł priorytetowych P_0–P_{11}. Zastosowanie większości z reguł priorytetowych wyznaczanych na podstawie analizy danych projektowych P_1–P_{11} daje lepsze rezultaty niż zastosowanie reguły losowych priorytetów P_0 (można to zaobserwować zwłaszcza przy harmonogramowaniu w przód). Najlepsze zasady ustalania kolejności zadań przy harmonogramowaniu w przód to minimalny najpóźniejszy czas rozpoczęcia (P_2) oraz minimalny najpóźniejszy czas zakończenia zadań (P_3), natomiast przy harmonogramowaniu wstecz minimalny najwcześniejszy czas zakończenia zadań (P_4) przy szeregowej procedurze dekodującej SGS. Występuje zróżnicowanie osiąganych rezultatów w zależności od przyjętej reguły priorytetowej.

Największą liczbę najlepszych uszeregowań, zarówno dla instancji testowych ze zbioru J30 (336 harmonogramów optymalnych spośród 480 możliwych) jak i J90 (330

harmonogramów spośród 480 możliwych o czasie trwania równym najlepszemu znanemu rozwiązaniu), znaleziono przy zastosowaniu heurystyki wieloprzebiegowej z regułą priorytetową P_4 (minimalny najwcześniejszy czas zakończenia zadań), harmonogramowaniu wstecz z szeregową procedurą SGS. Najgorsze wyniki dla większości reguł priorytetowania zadań są osiągane dla harmonogramowania wstecz z równoległą procedurą dekodującą SGS.

6. Podsumowanie

W artykule przedstawiono algorytmy priorytetowe (jedno- i wieloprzebiegowy) dla problemu harmonogramowania projektu z minimalizacją czasu trwania projektu. Wyniki eksperymentów obliczeniowych potwierdziły skuteczność opracowanej przez autorów heurystyki wieloprzebiegowej. Wskazały też na skuteczne reguły priorytetowe dla rozważanego zagadnienia. Najskuteczniejsza w poszukiwaniu harmonogramów optymalnych (dla zbioru instancji testowych J30) i identycznych z najlepszymi znanymi (dla zbioru instancji testowych J90) okazała się heurystyka wieloprzebiegowa z reguła priorytetową minimalny najwcześniejszy czas zakończenia zadań przy harmonogramowaniu wstecz i szeregowej procedurze dekodującej.

Proponowane heurystyki działające na podstawie reguł priorytetowych są efektywne i szybkie w działaniu. Mogą być wykorzystane jako rozwiązania startowe w algorytmach metaheurystycznych tj. symulowane wyżarzanie.

Literatura

1. Błażewicz J., Lenstra J., Kan A. R.: Scheduling subject to resource constraints - classification and complexity. Discrete Applied Mathematics, 5, 1983, 11-24.
2. Hartmann S., Briskorn D.: A Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem. European Journal of Operational Research, 207(1), 2012, 1-14.
3. Hartmann S., Kolisch R., Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Re-search, 127, 2000, s. 394-407.
4. Klimek M.: Predyktywno-reaktywne harmonogramowanie produkcji z ograniczoną dostępnością zasobów. Praca doktorska, AGH Kraków, 2010.
5. Klimek M., Łebkowski P.: Algorytm priorytetowy harmonogramowania projektu przy ograniczonych zasobach, [w:] Komputerowo Zintegrowane Zarządzanie, Knosala R. (red.), Opole: Oficyna Wydawnicza Polskiego Towarzystwa Zarządzania Produkcją, T.I, 2008, s. 532-540.
6. Kolisch R.: Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, European Journal of Operational Research, 90, 1996, 320-333.
7. Kolisch, R. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operations Management 14, 1996, s. 179–192.
8. Kolisch R., Hartmann S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update, European Journal of Operational Re-search 174, 2006, s. 23-37.

9. Kolisch R., Padman R.: An integrated survey of deterministic project scheduling, OMEGA The International Journal of Management Science, 29, 2001, 249-272.
10. Kolisch R., Sprecher A.: PSPLIB – a project scheduling library. European Journal of Operational Research 96, 1997, s. 205–216.
11. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne, Wydawnictwo WNT, Warszawa, 1999

Dr inż. Marcin KLIMEK
Zakład Informatyki, Katedra Nauk Technicznych,
Wydział Nauk Ekonomicznych i Technicznych,
Państwowa Szkoła Wyższa im. Papieża Jana Pawła II
21-500 Biała Podlaska,
ul. Sidorska 95/97,
e-mail: m.klimek@dydaktyka.pswbp.pl

Dr hab. inż. Piotr ŁEBKOWSKI, prof. AGH
Katedra Badań Operacyjnych i Technologii Informacyjnych,
Wydział Zarządzania AGH
30-067 Kraków,
ul. Gramatyka 10,
e-mail: plebkows@zarz.agh.edu.pl